

Real6410 Android 2.1 Development manual

Ver 1.6

Date: 2010-11-20

Change History

Rev	Date	Description
V1.0	2010-5-20	The initial released Version
V1.1	2010-10-7	Change the cross compile install erro
V1.2	2010-10-20	Android system update, for gprs, 2D function
V1.6	2010-11-20	update the download method

Catalogue

Real6410 Android 2.1 Development manual	1
Catalogue	3
Chapter 1 Android 2.1 Overview	5
Chapter 2 Android burning	6
2.1 Burning bootloader	6
2.1.1 Create SD boot card	6
2.1.2 Burning the image into flash by SD card	7
2.2 Burning Linux kernel	10
2.3 Burning cramfs	10
2.4 Burning Android system	11
Chapter 3 Uboot compile	14
3.1 Uboot configure	14
3.2 Uboot compile	14
Chapter 4 Linux kernel compile	15
4.1 kernel compile	15
4.2 Kernel configure	15
4.2.1 Touchscreen configure	16
4.2.1 Keyboard configure	16
4.2.3 Audio configure	17
4.2.4 MMC/SD	18
4.2.5 Nandflash configure	20
4.2.6 LCD configure	20
4.2.7 Etherne configure	21
4.2.8 WIFI configure	21
4.2.9 usb adb configure	22
4.2.10 Camera	23
4.2.11 jpeg configure	25
4.2.12 GPS configure	26
Chapter V Android System Development	27
5.1 Build Development environment	27
5.1.1 Install essential packages	27
5.1.2 Install JDK 5.0	27
5.2 Android compile	28
5.2.1 compile Android	28
5.2.2 Android module support	28
Chapter VI Android function Use	29
6.1 Ethernet use	29
6.2 USB adb in Android 2.1	29
6.3 Camera use in android 2.1	30
6.4 WiFi use in android 2.1	31

6.5 Switch horizontal and vertical screen	32
6.6 Dynamic Wallpapers	32
6.7 Install APK file in android 2.1	32
6.8 GPS module use.....	33
Appendix A DNW software configuration	34
Appendix B USB driver install	35
Appendix C Cross compile tools install	38
C-1 Uncompress the tools	38
C-2 Add Path in your environment file	38
C-3 Check the tool-chain path to see if it is set up correctly or not.	38

Chapter 1 Android 2.1 Overview

Boot loader

- version: s3c-u-boot-1.1.6
- Function: support boot and update system by SD card and USB

Linux kernel

- version: s3c-Linux-2.6.28.4
- Compile: jdk5
- Function: **support Jpeg encode, USB adb debug, AVD support**

Device Driver

- TFT LCD/Touchscreen, Audio OUT, MMC/SD card, NET, USB OTG, Serial port
- watchdog, RTC, keyboard
- **Support WIFI, GPS, Camera driver**

File System support

- Ubi filesystem

Function use

- **Ethernet**
- **USB adb debug**
- **Switch horizontal and vertical screen**
- **Dynamic Wallpapers**
- **APK program install method**
- **WIFI use**
- **Camera use**
- **GPS module use**

Chapter 2 Android burning

The Android image contains 3 parts, namely bootloader, kernel, rootfs, it is the same with the Linux system, this chapter will introduce the method to burning the image to the board.

The Address for the parts was as follow:

Name	Address Range	details
bootloader	0x00000000~0x0003FFFF	for uboot image
kernel	0x00040000~0x003FFFFFFF	for linux kernel image
cramfs	0x00400000~0x007FFFFFFF	for cramfs format filesystem(download ubifs)
Ubifs	0x00800000~0x3FFFFFFF	for ubifs format filesystem

Preparations for burning:

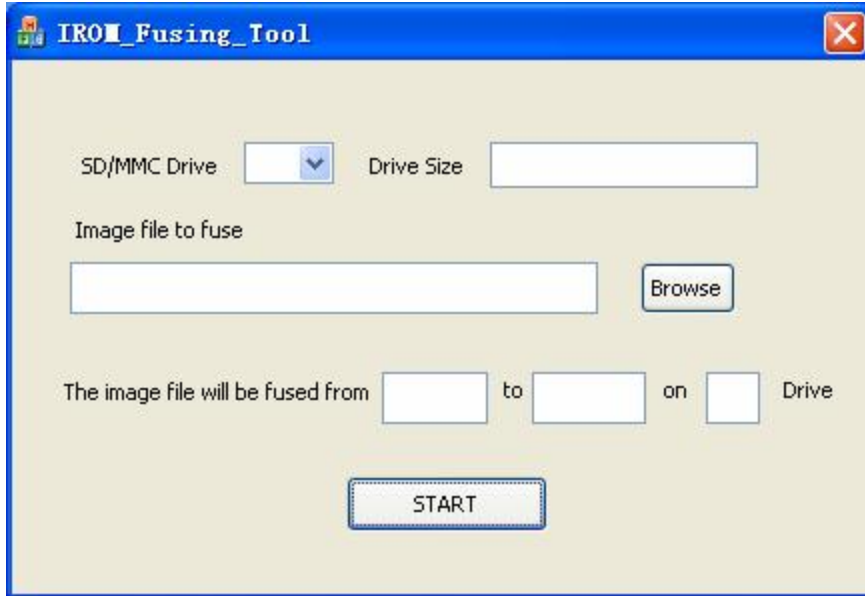
- link the serial line from PC to the board UART0,
- link the Power to the board
- link the usb cable from PC to the board
(About the board interface, please refer to the **Appendix B Board interface of the <hardware user manual>**)
- Open the DNW software, and configure it(refer to the [Appendix A DNW software configuration](#))

2.1 Burning bootloader

2.1.1 Create SD boot card

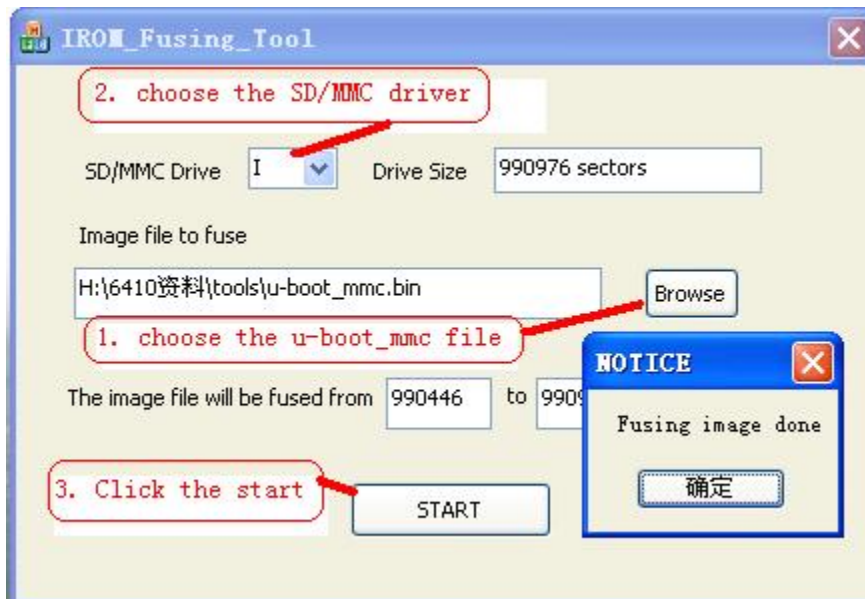
(1) Insert the SD card to USB reader under WinXP, and format the SD card to FAT32 format.

(2) Run the **IROM_Fusing_Tool.exe** tools,
the tools position:\tools\SDboot\IROM_Fusing_Tool.exe



3) burn bootloader

- Click “**Browse**”, add the file **uboot_mmc.bin**,
The file position: **\tools\SDboot\uboot_mmc.bin**
- select SD card in **SD/MMC Drive** under tools.
- Click “**START**”



After burning the image successfully, there will be a pop-up windows “Fusing image done”, Click “Ok” to finish creating the SD card.

2.1.2 Burning the image into flash by SD card

1) Insert the SD card to the Real6410.

2) Set the board for SD boot mode

- set the digital switch to boot from SD mode as follow:

boot mode / Pin	1	2	3	4	5	6	7	8
SD card boot	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF
Nand flash boot mode	OFF	OFF	OFF	OFF	OFF	ON	ON	OFF

3) Power on the board, then the DNW will print the message,

```

DNW v0.60C - For WinCE [COM4, 115200bps] [USB:x] [ADDR:0xc000000]
Serial Port USB Port Configuration Help
U-Boot 1.1.6 (Mar 3 2010 - 19:43:11) for SMDK6410

CPU: S3C6410@800MHz 800M
Fclk = 800MHz, Hclk = 133MHz, Pclk = 66MHz, Serial = CLKUART (SYNC
Mode)
Board: SMDK6410
DRAM: 256 MB 256MB Memory
Flash: 0 kB
NAND: Maf. ID is d3 1024 MB 1G nand flash
MMC: => rca=0x0000e624 484 MB 512M SD Card
Command NOT Complete Movinand means the sd card boot
*** Warning - bad CRC or MovinAND, using default environment

In: serial
Out: serial
Err: serial

Hit any key to stop autoboot: 0
SMDK6410 #

```

4) Then within 3 second, press “Space” Key on PC keyboard, enter **BOOT** command line.

SMDK6410 #

5) Format nand flash, run the follow command in the **BOOT** command line.

SMDK6410 # nand erase 0

Notice: If it have also the ecc error, run the follow command, or don't run it.

SMDK6410 # nand scub

6) Run **dnw** in te **BOOT** command line

SMDK6410 # dnw

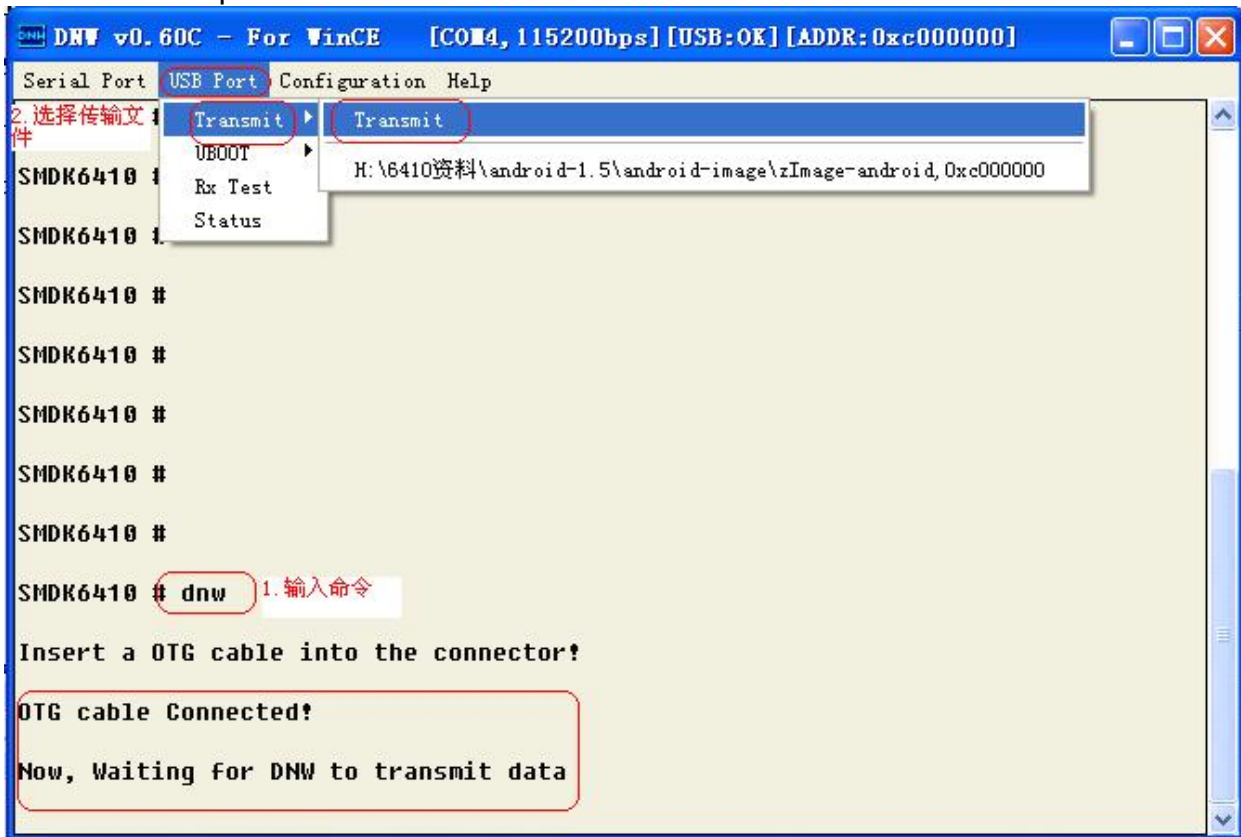
Notice: It need the usb download driver, if you don't install it, you should install it.

The USB download driver position: **CD\tools\usb driver\usb DNW driver**

You can refer the chapter ([Appendix B USB driver install](#)) to install it

7) In the DNW menu, Click "**USB Port->Transmit-> Transmit**", then choose **uboot.bin** file.

u-boot.bin file position: **CD\tools\SDboot\u-boot.bin**



8) run the command in **BOOT** command line

SMDK6410 # nand erase 0 40000

SMDK6410 # nand write c0008000 0 40000

Then the uboot will be burned in the Nand flash.

2.2 Burning Linux kernel

- set the digital switch to boot from Nand boot mode as follow:

boot mode / Pin	1	2	3	4	5	6	7	8
SD card boot	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF
Nand flash boot	OFF	OFF	OFF	OFF	OFF	ON	ON	OFF

1) Power on the board, enter the BOOT command line in DNW, run the follow command:

```
SMDK6410 # dnw
```

2) In the DNW menu, Click "**USB Port->Transmit-> Transmit**", then choose **zImage_43_net** file.

zImage file position: **CD\Android\Image\zImage_43_net**

Notice: the different name have the different LCD

zImage_43_net --> 4.3"LCD can support net

zImage_43_gprs --> 4.3"LCD can support gprs

and the 70 means 7"LCD, choose the image for your lcd size.

3) run the command in **BOOT command line**

```
SMDK6410 # nand erase 40000 300000
```

```
SMDK6410 # nand write c0008000 40000 300000
```

Then the Linux kernel will be burned in the Nand flash.

2.3 Burning cramfs

The cramfs burning is only for burning the Android system.

1) enter the BOOT command line in DNW, run the follow command:

```
SMDK6410 # dnw
```

2) In the DNW menu, Click "**USB Port->Transmit-> Transmit**", then choose **rootfs.cramfs** file.

rootfs.cramfs file position: **CD\Android\Image**

3) run the command in **BOOT command line**

```
SMDK6410 # nand erase 400000 400000
```

```
SMDK6410 # nand write c0008000 400000 400000
```

Then the Linux cramfs will be burned in the Nand flash.

4) Test the cramfs

run the command in **BOOT command line:**

```
SMDK6410 # setenv bootargs noinitrd root=/dev/mtdblock0 console=ttySAC0 init=/linuxrc
SMDK6410 # saveenv
```

Then reboot the board, If it show as that, it means the cramfs is ok.

```
Serial Port USB Port Configuration Help
7>ieee80211_crypt: registered algorithm 'NULL'
<7>ieee80211_crypt: registered algorithm 'WEP'
<7>ieee80211_crypt: registered algorithm 'CCMP'
<7>ieee80211_crypt: registered algorithm 'TRIP'
<6>UFP support v0.3: UFP support v0.3: implementor 41 architecture 1 part 20
variant b rev 5
implementor 41 architecture 1 part 20 variant b rev 5
<6>s3c2410-rtc s3c2410-rtc: setting system clock to 2030-03-03 23:53:06 UTC
(1898812386)
s3c2410-rtc s3c2410-rtc: setting system clock to 2030-03-03 23:53:06 UTC
(1898812386)
UFS: Mounted root (cramfs filesystem) readonly.
UFS: Mounted root (cramfs filesystem) readonly.
<6>Freeing init memory: 276K
Freeing init memory: 276K
ifconfig: SIOCSIFADDR: No such device
*****Main Menu*****
Use this fs to make real fs.
0.First of all,we need to format the mtdblock1 to ubifs.
a.Yes,do it now
b.No,next time|
```

2.4 Burning Android system

Make sure the Cramfs is boot ok, If it is not ok, Please burn it firstly refer the [2.3 Burning cramfs](#).

1) copy the **android.tar.gz** file to the SD card or usb disk, and link the Sd card or usb disk to the board.

Notice: the different name have the different LCD

android_43_2D.tar

android_43_no_2D.tar

android_70_2D.tar

android_70_no_2D.tar

It have four android image in the folder, you can choose the right on as follow:

43 means 4.3"LCD

70 means 7"LCD

2D means support 2D in android.

no_2D means don't support 2D in android.

Choose one file you need, can copy it to sd card, then change the name for android.tar.gz

2) Power on the board, boot the linux and enter the cramfs system,

3) input "a" and press "Enter" in the PC, then it will show as that:

```

DNU v0.60C - For WinCE [COM4, 115200bps] [USB:x] [ADDR: 0xc000000]
Serial Port USB Port Configuration Help
UBIFS: default file-system created
<5>UBIFS: mounted UBI device 0, volume 0, name "rootfs"

UBIFS: mounted UBI device 0, volume 0, name "rootfs"
<5>UBIFS: file system size: 249532416 bytes (243684 KiB, 237 MiB, 967 LEBs)

UBIFS: file system size: 249532416 bytes (243684 KiB, 237 MiB, 967 LEBs)
5>UBIFS: journal size: 12386304 bytes (12096 KiB, 11 MiB, 48 LEBs)

UBIFS: journal size: 12386304 bytes (12096 KiB, 11 MiB, 48 LEBs)
5>UBIFS: media format: 4 (latest is 4)

UBIFS: media format: 4 (latest is 4)
<5>UBIFS: default compressor: LZ0

UBIFS: default compressor: LZ0
<5>UBIFS: reserved for root: 5182151 bytes (5060 KiB)

UBIFS: reserved for root: 5182151 bytes (5060 KiB)
1.Put your rootfs tar ball in sdcard or udisk,then,insert it
***For android,the tar name should be android.tar.gz.**
***For qtopia,the tar name should be qtopia.tar.gz.***
Now pls Select the system you want to creat
a.Android
b.Qtopia
c.Exit

```

4) input "a" and press "Enter" in the PC to burning the Android auto.

```

DHW v0.60C - For WinCE [COM4, 115200bps] [USB:x] [ADDR:0xc000000]
Serial Port USB Port Configuration Help
7>ieee80211_crypt: registered algorithm 'NULL'
<7>ieee80211_crypt: registered algorithm 'WEP'
<7>ieee80211_crypt: registered algorithm 'CCMP'
<7>ieee80211_crypt: registered algorithm 'TRIP'
<6>UFP support v0.3: UFP support v0.3: implementor 41 architecture 1 part 20
variant b rev 5
implementor 41 architecture 1 part 20 variant b rev 5
<6>s3c2410-rtc s3c2410-rtc: setting system clock to 2030-03-03 23:53:06 UTC
(1898812386)
s3c2410-rtc s3c2410-rtc: setting system clock to 2030-03-03 23:53:06 UTC
(1898812386)
UFS: Mounted root (cramfs filesystem) readonly.
UFS: Mounted root (cramfs filesystem) readonly.
<6>Freeing init memory: 276K
Freeing init memory: 276K
ifconfig: SIOCSIFADDR: No such device
*****Main Menu*****
Use this fs to make real fs.
0.First of all,we need to format the mtdblock1 to ubifs.
a.Yes,do it now
b.No,next time|
Boot success

```

- 5) If it print "**Do you want to reboot now(y/n)**", it means download success.
- 6) reboot the board, press "Space" Key on PC keyboard, enter **BOOT command line**
SMDK6410 #
- 7) run the command in **BOOT command line**:
SMDK6410 #
setenv bootargs noinitrd console=ttySAC0 init=/init ubi.mtd=1 root=ubi0:rootfs rootfstype=ubifs fbcon=rotate:1 mem=224M
SMDK6410 # saveenv
- 8) Then reboot the board, it will boot into the android system automatic.

Chapter 3 Uboot compile

You should install the cross compile tools(arm-none-linux-gnueabi-4.3.2 with EABI) before you compile the uboot, if you don't install it, please install it refer to the [Appendix C Cross compile tools install](#)

3.1 Uboot configure

copy the file s3c-u-boot-1.1.6-Real6410.tar.bz2 From CD to ubuntu system in PC.
s3c-u-boot-1.1.6-Real6410.tar.bz2 position: **CD\linux\code**

```
# mkdir bootloader
# cd bootloader
# cp -a /media/cdrom/linux/code/s3c-u-boot-1.1.6-Real6410.tar.bz2 .
# tar xvf s3c-u-boot-1.1.6-Real6410.tar.bz2
# cd s3c-u-boot-1.1.6-Real6410
```

Then we can get the **s3c-u-boot-1.1.6-Real6410** folders in bootloader folder, run the command to configure it:

```
# make distclean
# make smdk6410_config
```

3.2 Uboot compile

We can get the uboot image for different boot mode image, SD card boot and Nand flash boot image.

Compile the uboot image for **Nand flash boot**:

```
# ./make_nand_image
```

If there is no error, u-boot.bin should have been generated with a file size of about 130~150Kbyte. you can get it in **bootloader/s3c-u-boot-1.1.6-Real6410** folder

Compile the uboot image for **SD card boot**:

```
#./make_mmc_image
```

If there is no error, u-boot_mmc.bin should have been generated with a file size of about 130~150Kbyte. you can get it in **bootloader/s3c-u-boot-1.1.6-Real6410** folder.

Chapter 4 Linux kernel compile

You should install the cross compile tools(arm-none-linux-gnueabi-4.3.2 with EABI) before you compile the uboot, if you don't install it, please install it refer to the [Appendix C Cross compile tools install](#)

4.1 kernel compile

copy the file s3c-linux-2.6.28.6-Real6410.tar from CD to ubuntu system in PC.

s3c-linux-2.6.28.6-Real6410.tar.bz2 position: **CD\linux\code**

```
# mkdir kernel
# cd kernel
# cp -a /media/cdrom/linux/code/s3c-u-boot-1.1.6-Real6410.tar.bz2 .
# tar xvf s3c-linux-2.6.28.6-Real6410.tar.bz2
# cd s3c-linux-2.6.28.6-Real6410
```

Then we can get the **s3c-linux-2.6.28.6-Real6410** folders in kernel folder, run the command to configure it:

```
# make distclean
# cp Real6410-android-43.config .config
```

Note: Real6410-android-43 was the 4.3"LCD configure file, if you use the other LCD, please use the right file.

```
Real6410-android-43.config      ----- 4.3 "LCD configure file
Real6410-android-50.config      ----- 5.0 "LCD configure file
Real6410-android-73.config      ----- 7.0"LCD configure file
```

Compile the kernel image :

```
# make
```

If you can build kernel successfully, you will have "arch/arm/boot/zImage" .

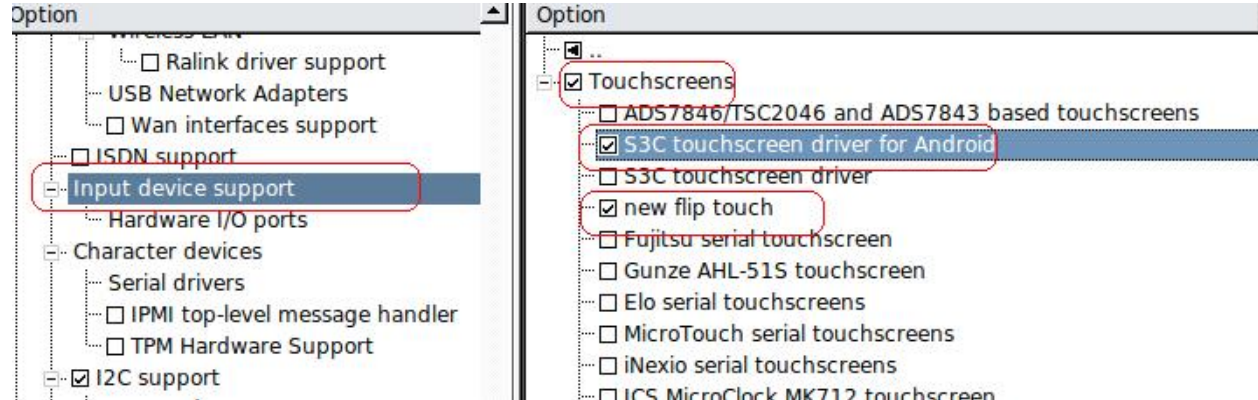
4.2 Kernel configure

If we want to configure the kernel driver by ourself, please refer the folow step:

we will use the **make xconfig** to configure the kernel, run the follow command to enter the configure menu.

```
# cd kernel/s3c-linux-2.6.28.6-Real6410
# make xconfig
```

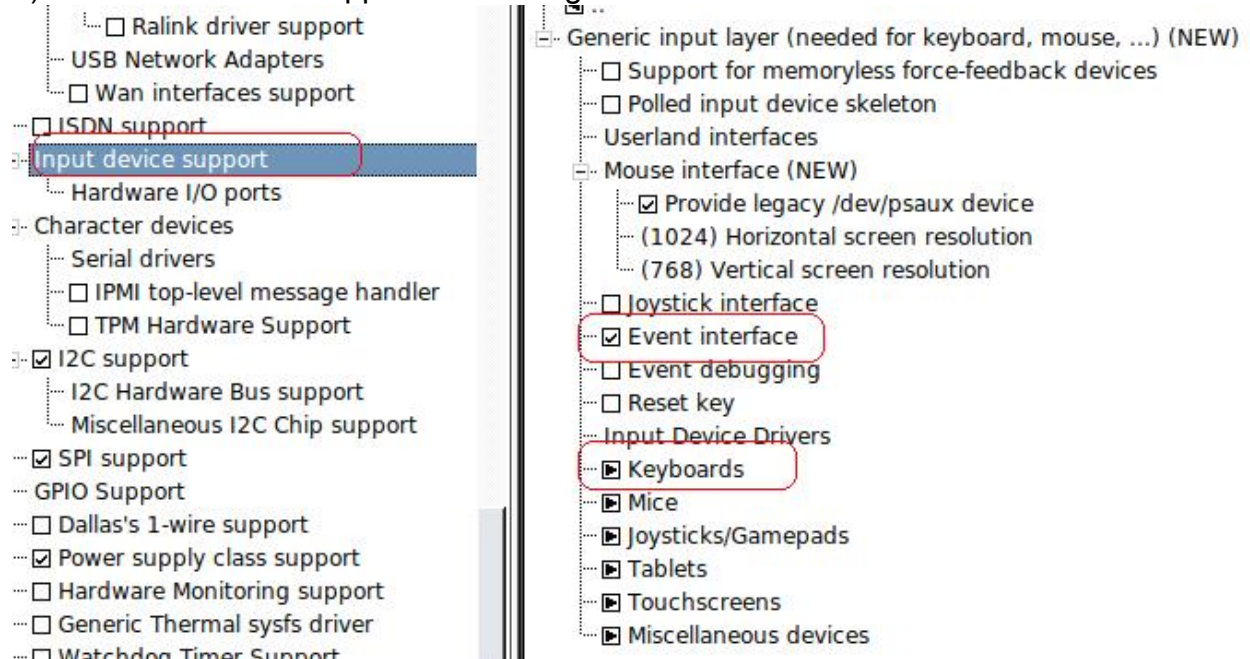
4.2.1 Touchscreen configure



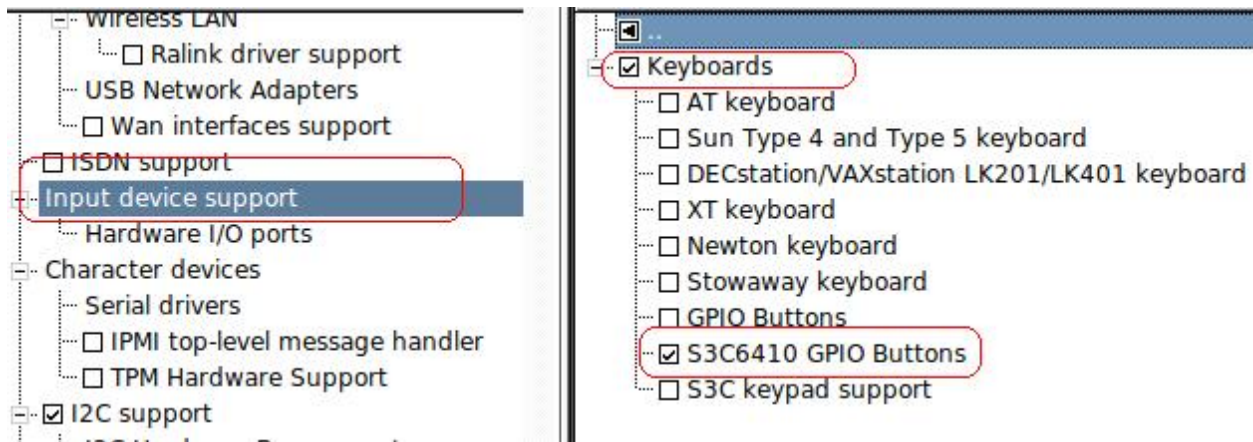
Touchscreen driver position: [drivers/input/touchscreen/s3c-ts_android.c](https://github.com/android/android_kernel_samsung_s3c6410/blob/master/drivers/input/touchscreen/s3c-ts_android.c)

4.2.1 Keyboard configure

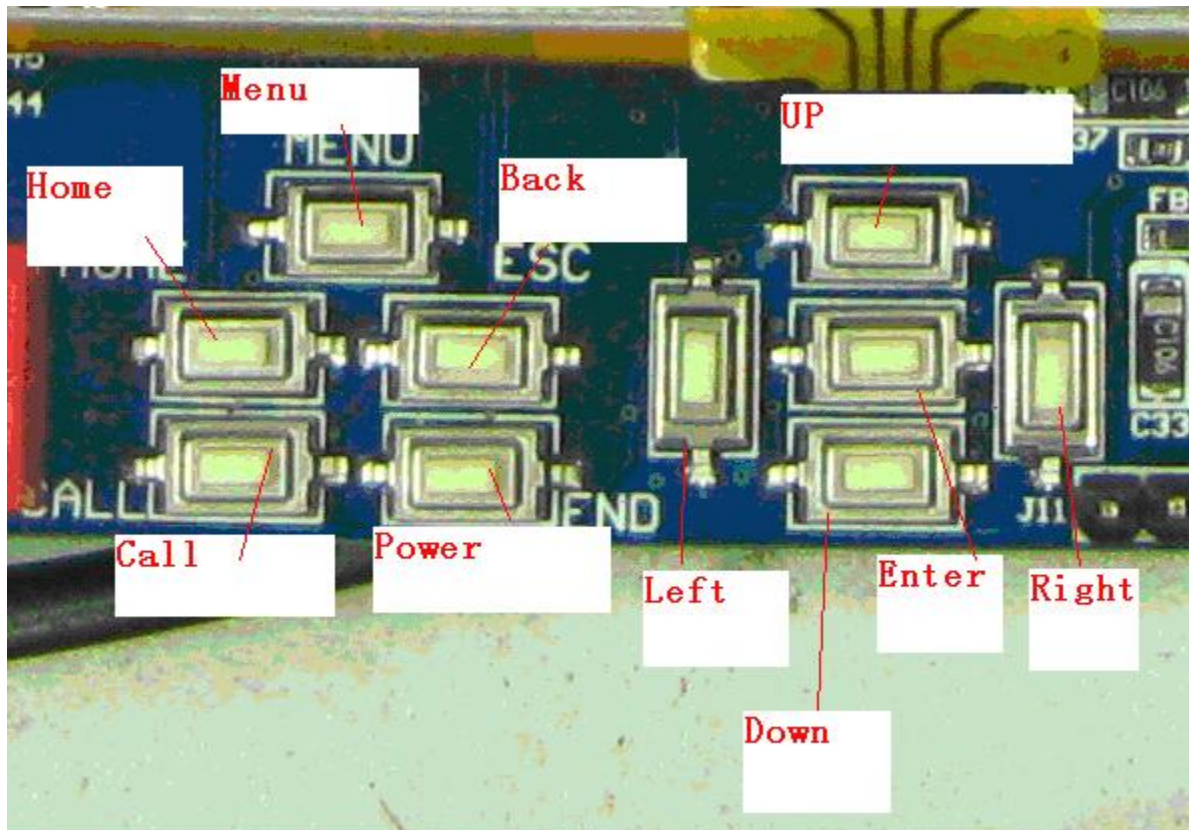
a) enter input device support then configure it.



b) enter the keyboards, configure



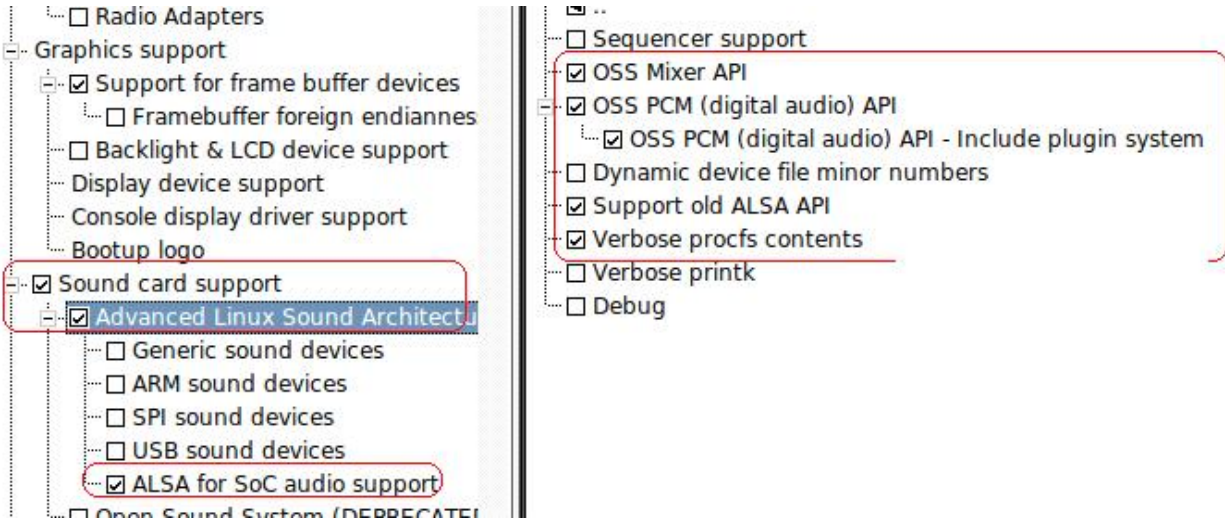
This is the button for android:



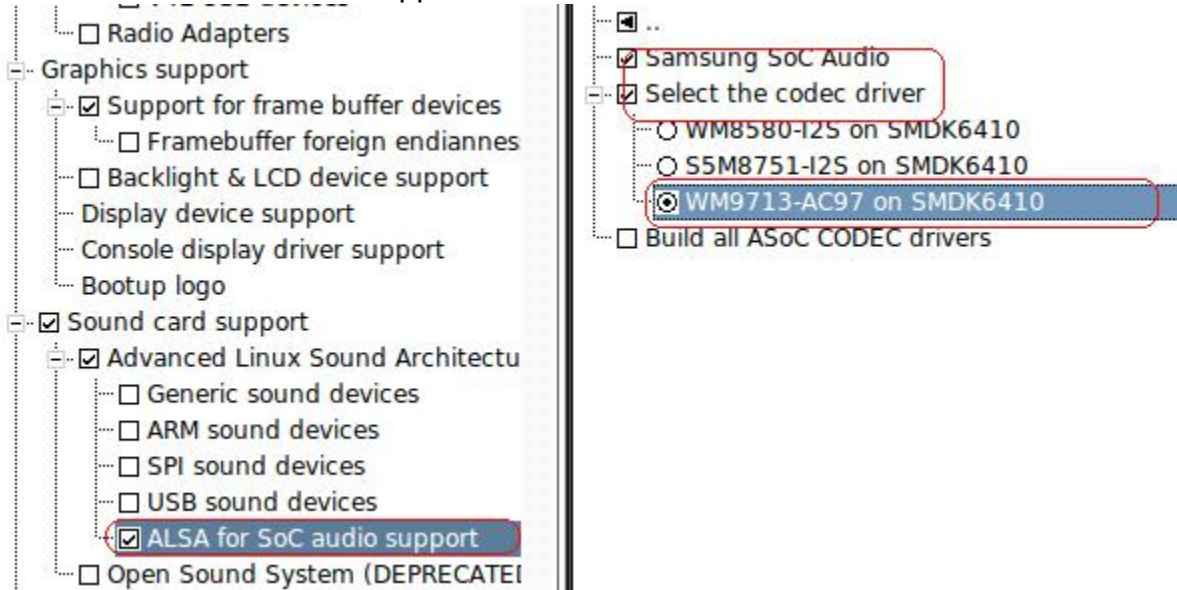
Keyboard driver position: [drivers/input/keyboard/ s3c-gpio_keys.c](#)

4.2.3 Audio configure

enter "Sound card support ", choose



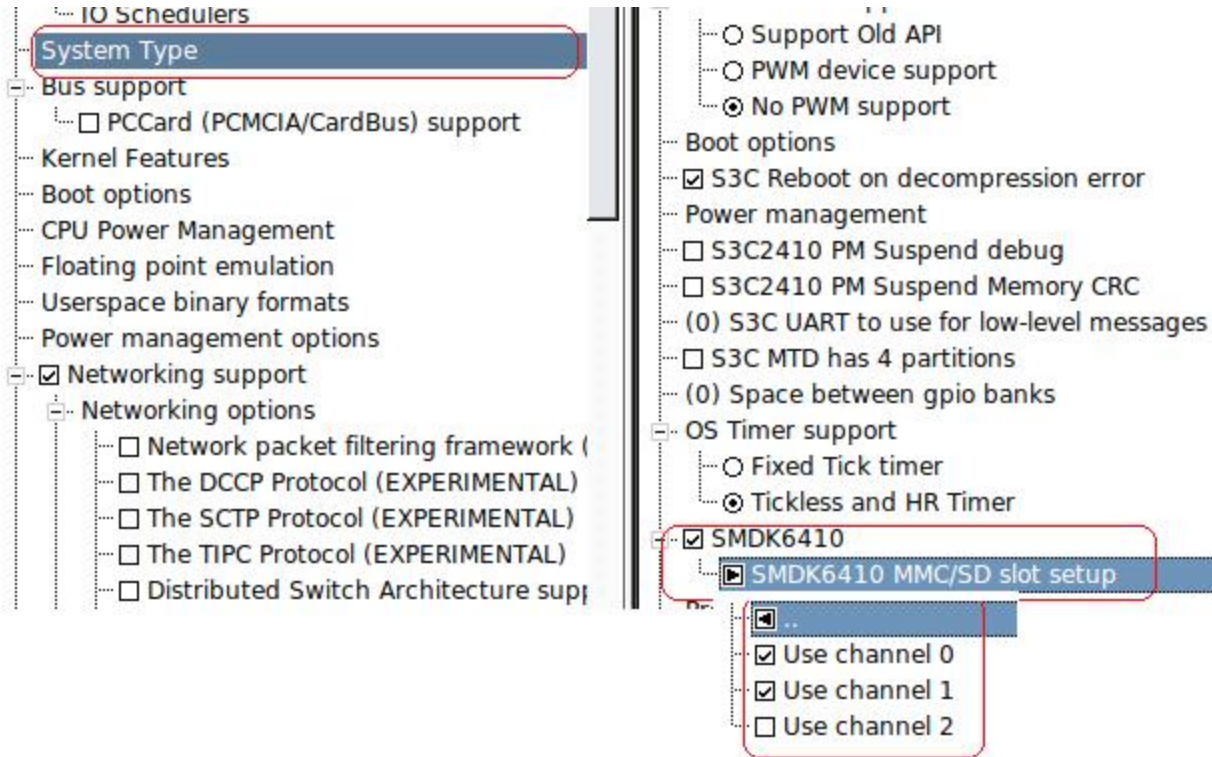
enter "ALSA for SoC audio support"



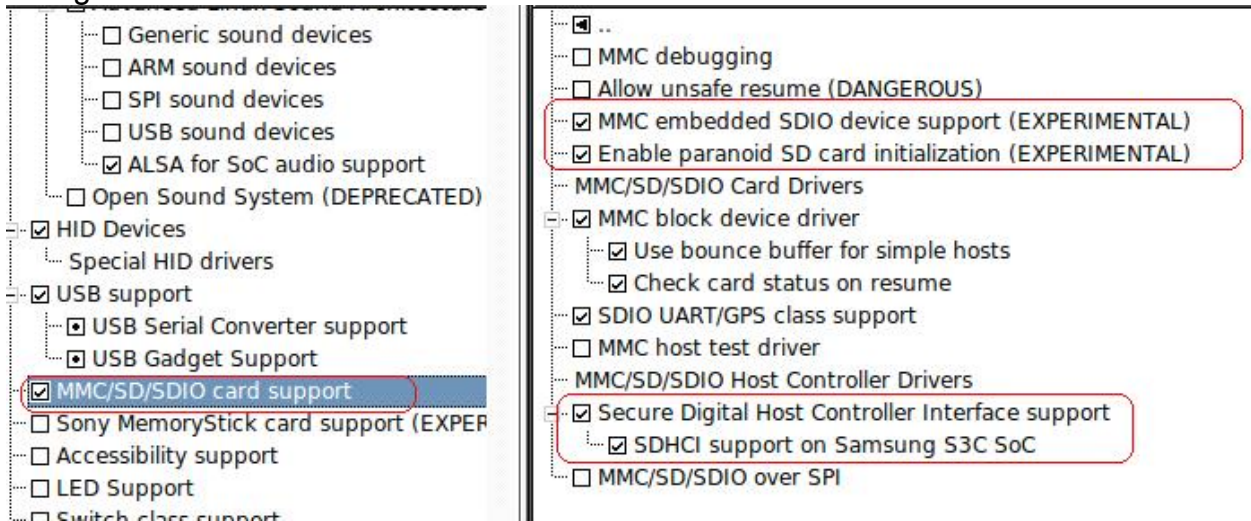
we use the WM9713 chips.
audio driver position: **sound/soc/s3c**

4.2.4 MMC/SD

choose the channel using in **System Type->SMDK6410->SMDK6410 MMC/SD slot Setup**

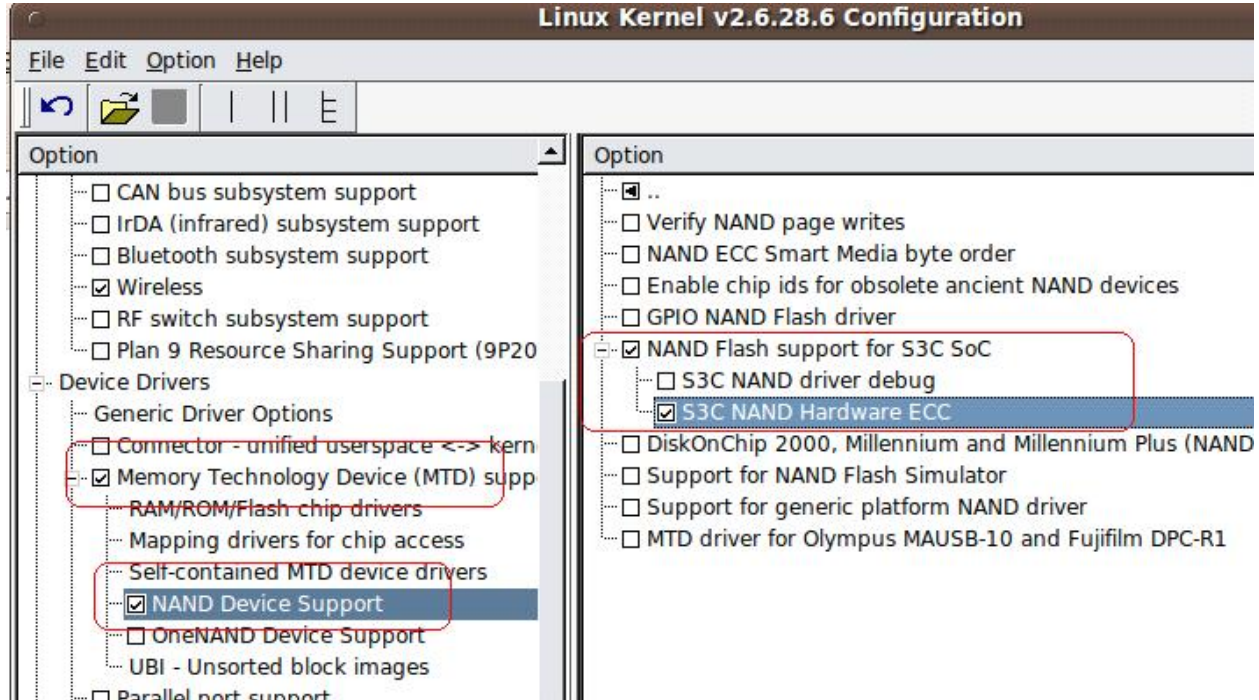


Choose **the channel 0** and **channel 1**, the channel 0 was used by wifi module, and the channel 1 was used by SD card. configure it as follow:



Mmc driver position: **drivers/mmc/host/sdhci-s3c.c**

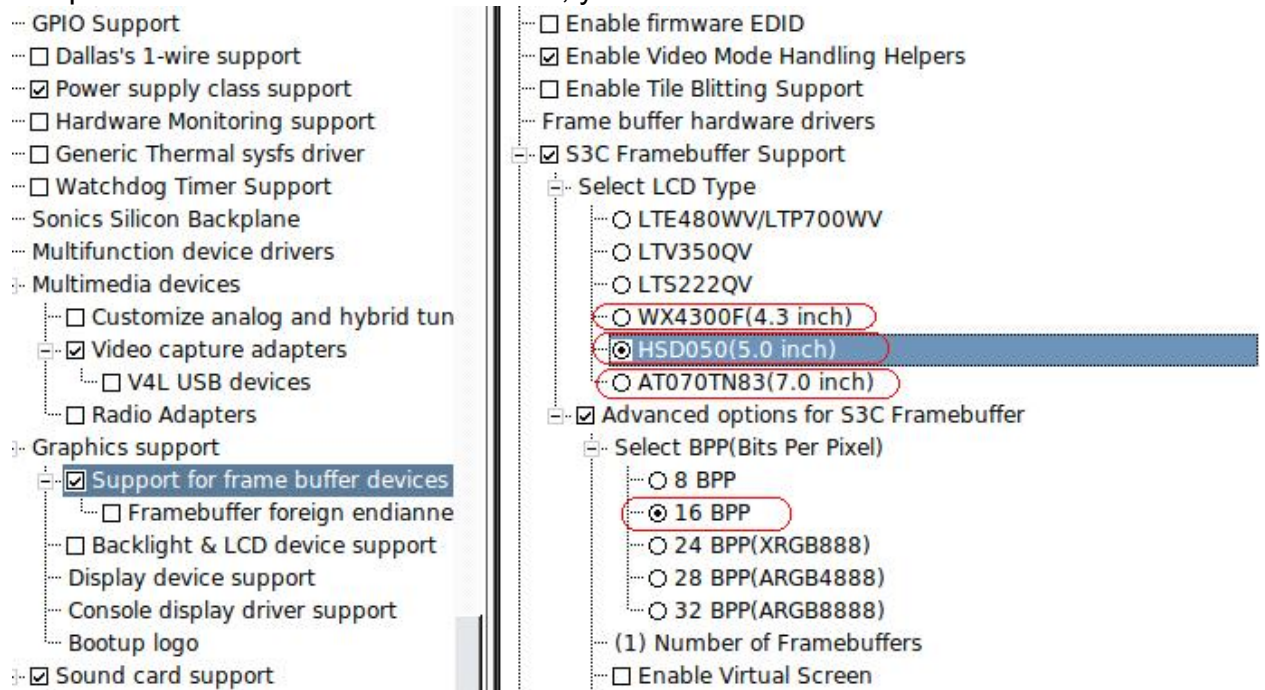
4.2.5 Nandflash configure



nand driver position: **drivers/mtd/nand/s3c_nand.c**

4.2.6 LCD configure

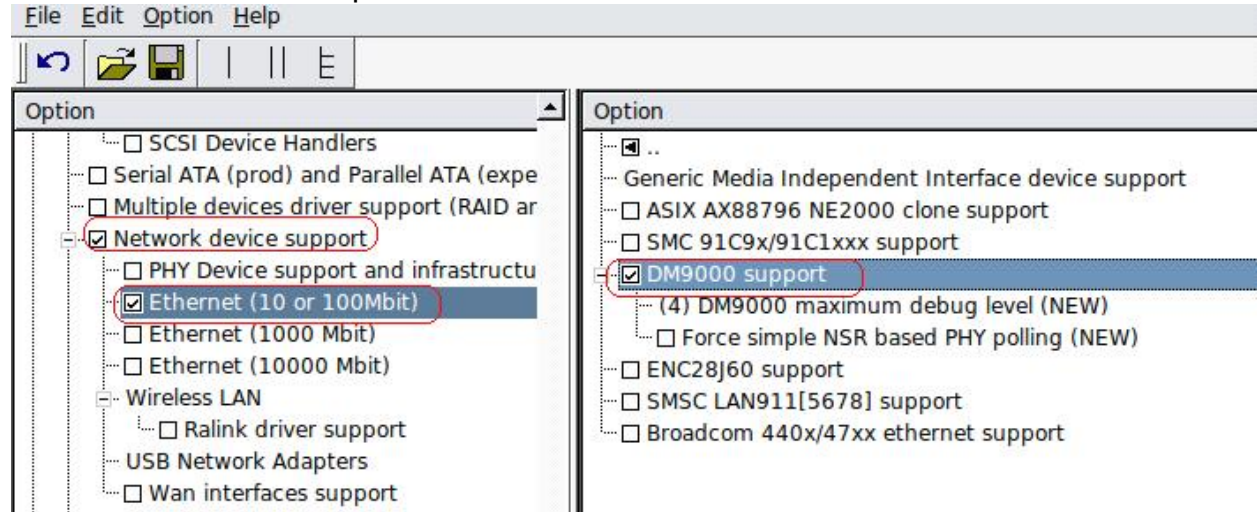
We provide different LCD for the board, you can choose it as :



LCD driver position: **drivers/video/samsung**

4.2.7 Etherne configure

We use the DM9000 chips in our board.

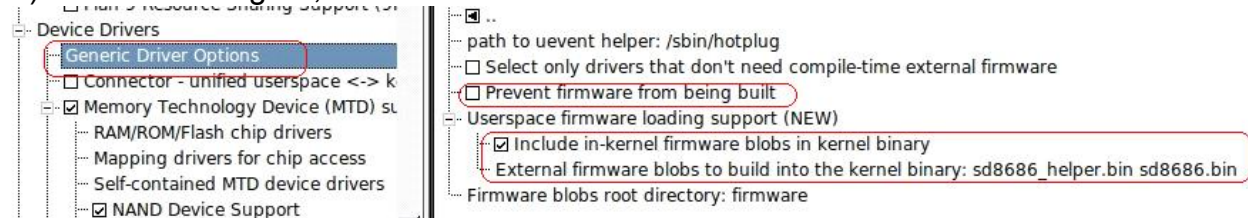


Ethernet positon: **drivers/net/dm9000.c**

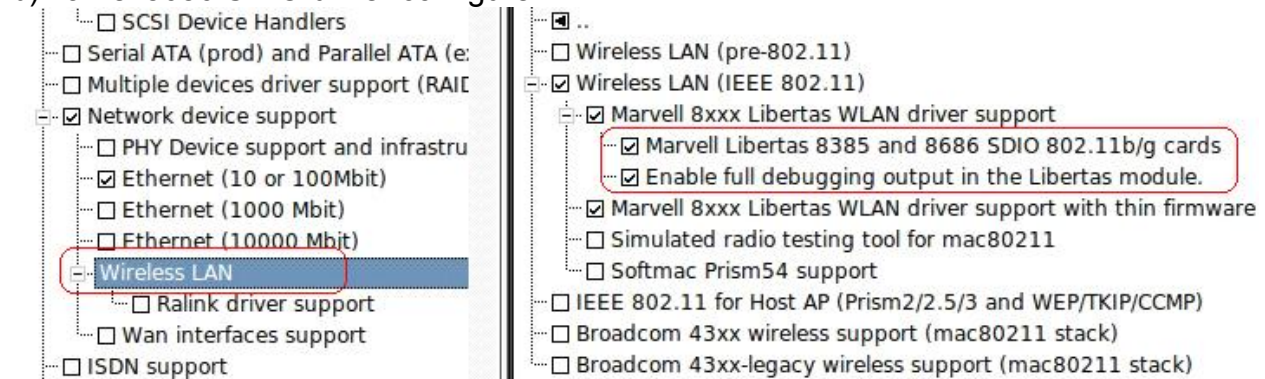
4.2.8 WIFI configure

we should configure two position, firmware and Marvel 8686 SDIO driver

a) Firewarc configure;



b)Marvel 8686 SDIO driver configure

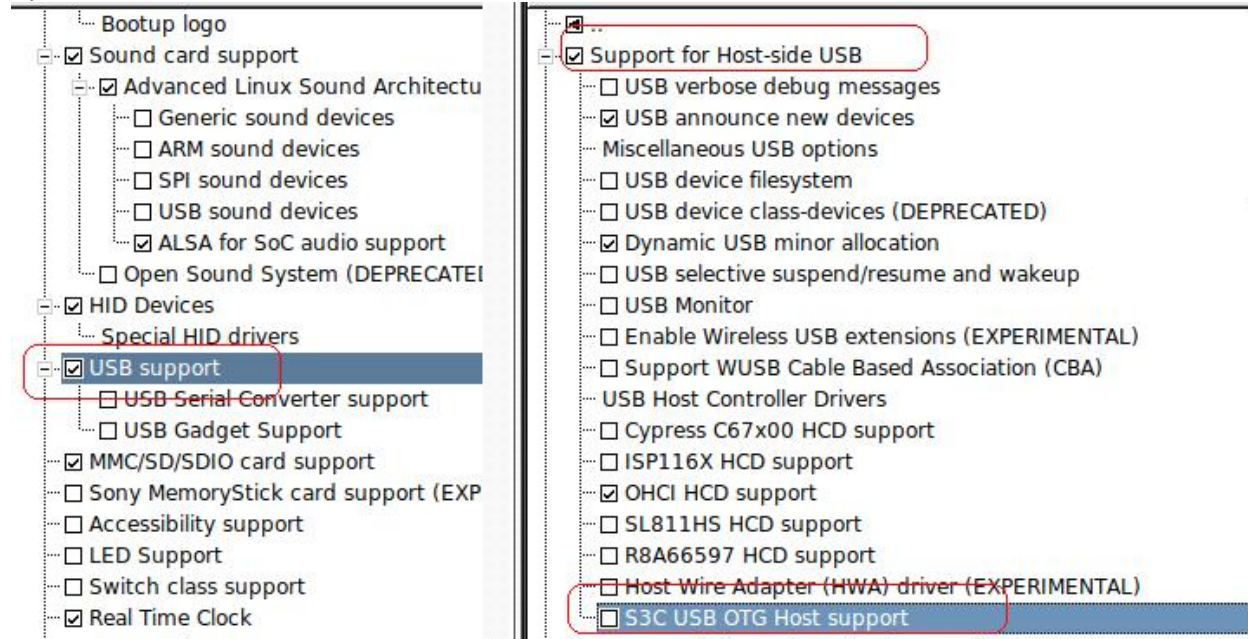


Wifi driver position: **drivers/net/wireless/libertas**

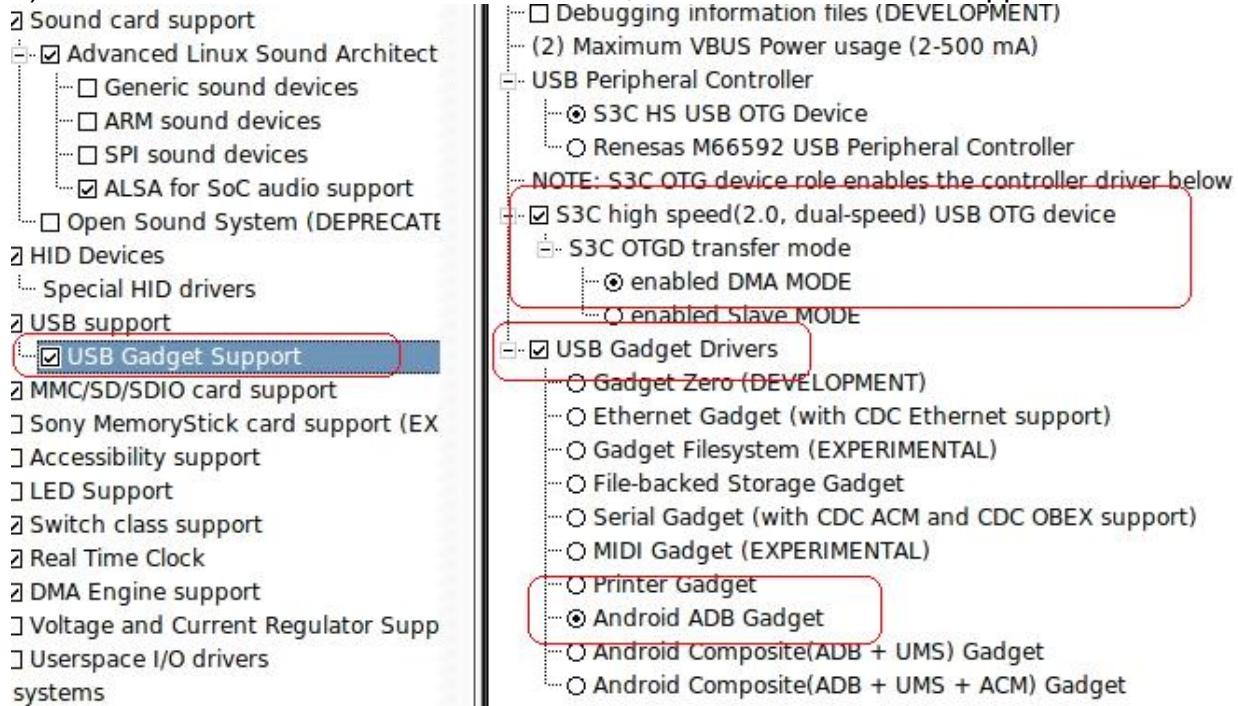
4.2.9 usb adb configure

The board can support the usb adb debug, configure is as follow:

a) remove the HOST function for USB OTG



b) choose the USB device function for OTG, and choose the adb support

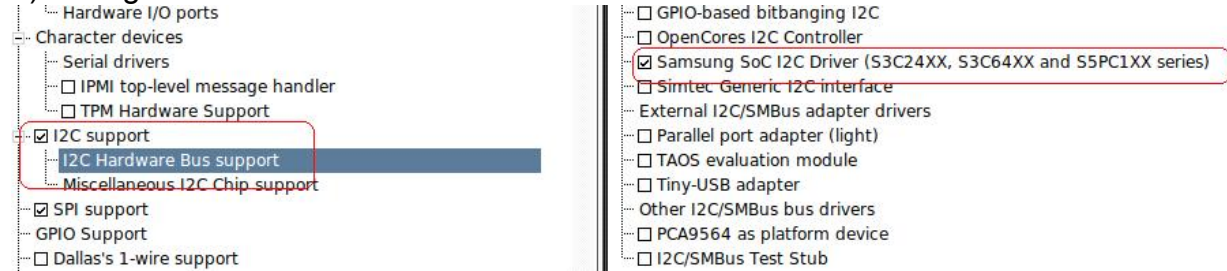


USB adb driver position: **drivers/usb/gadget/android_adb.c**

4.2.10 Camera

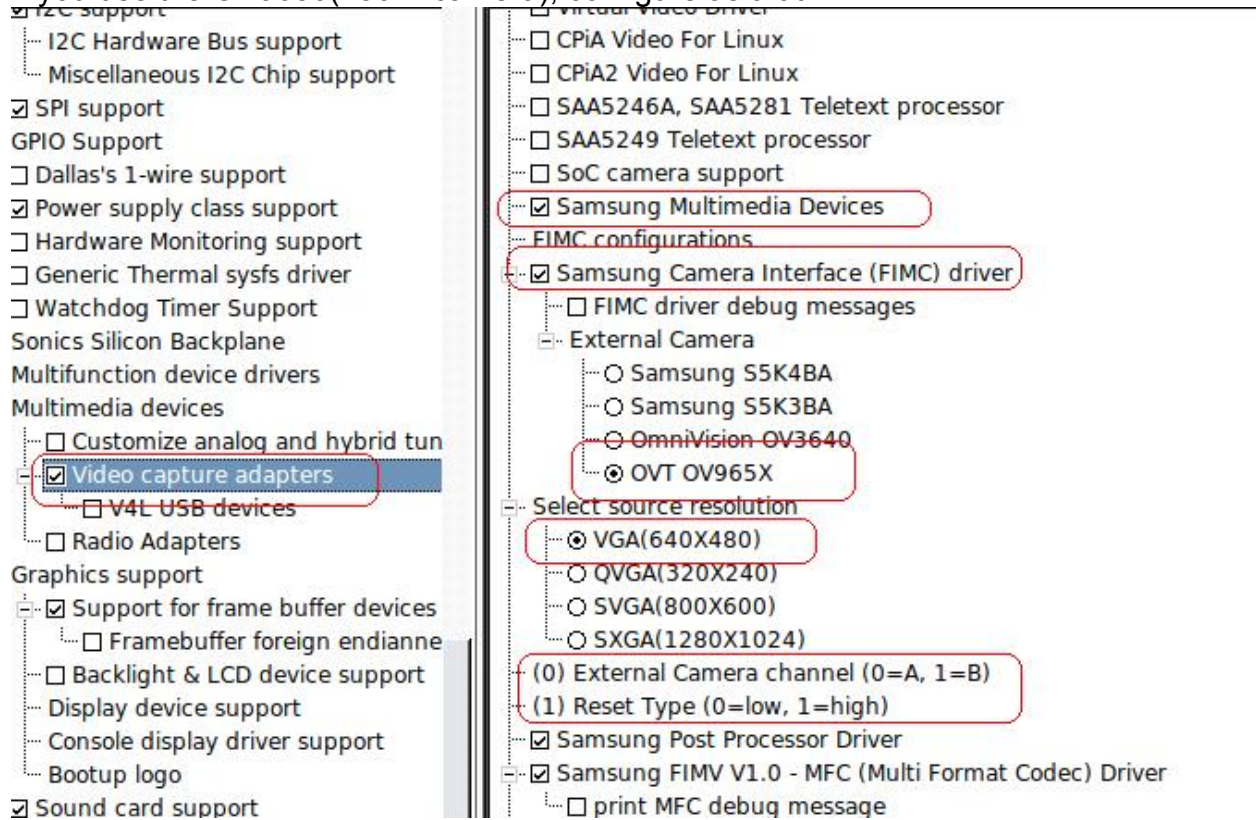
It support two camera module, OV9650(130W) and OV3640(300W), you can configure it by your module.

a) Configure the I2C

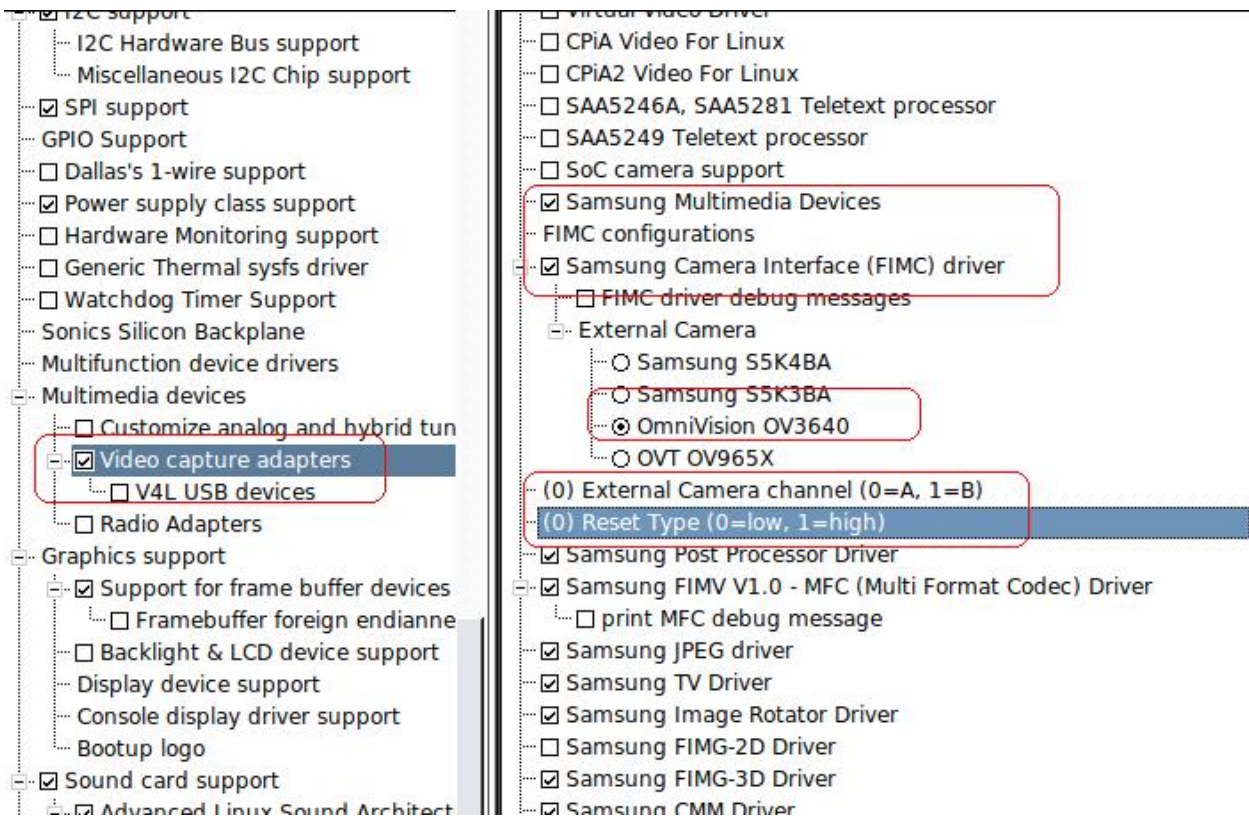


b) configure the Camera

If you use the OV9650(130W camera), configure as that:



If you use OV3640 (300W camera)



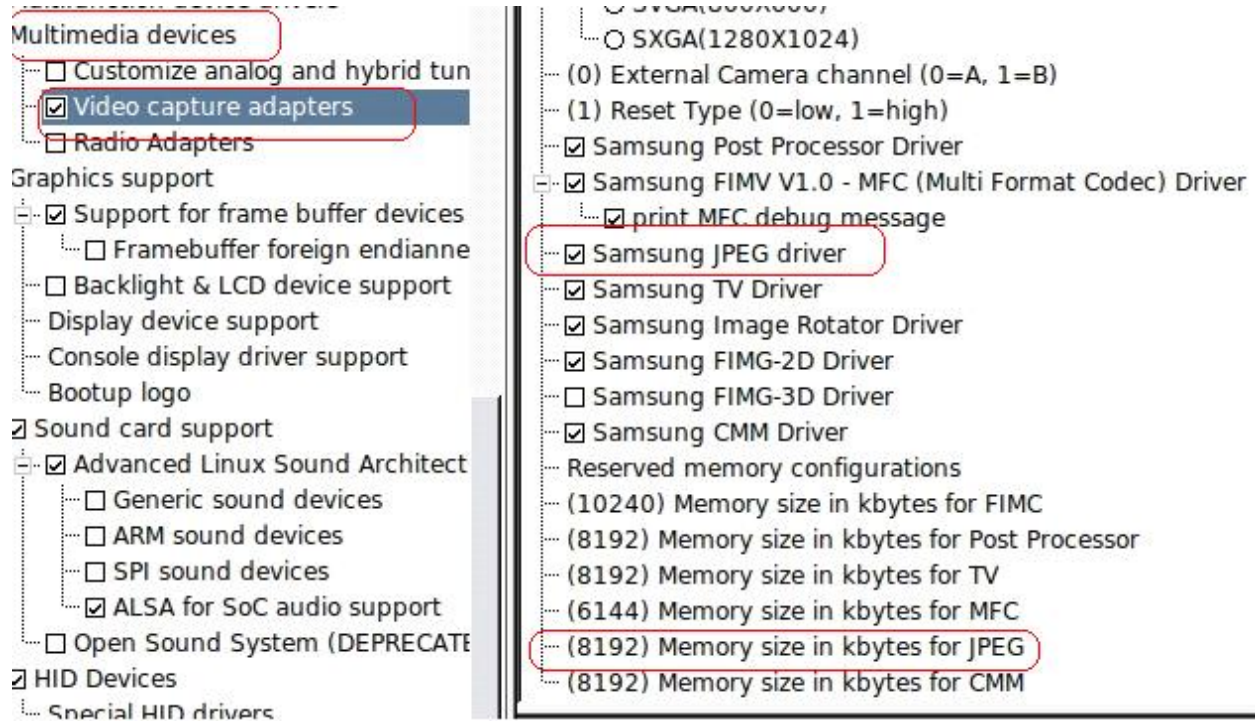
Note the camera insert method;



Camera driver position: **drivers/media/video/samsung/fimc**

4.2.11 jpeg configure

camera need the jpeg encode, it will encode the camera image to jpeg format.
configure it as that:



4.2.12 GPS configure

GPS use the uart2 to link the board, only the serial support is ok, so default the serial is ok, we don't need to configure it.

Chapter V Android System Development

Note: Compile the Android system, please use the Ubuntu system version, the other version may be have some problem for compiling.

5.1 Build Development environment

5.1.1 Install essential packages

Ubuntu need the follow essential packages for android

- flex: This lexical analyzer generator is used to read a given input file for a description of a scanner to generate.
- bison: This is a general-purpose parser generator.
- gperf: This is a perfect hash function generator.
- libesd0-dev: This enlightened sound daemon (dev files) is used to mix digitized audio streams for playback by a single device.
- libwxgtk2.6-dev: This package provides GUI components and other facilities for many different platforms.
- build-essential: This package contains a list of packages considered fundamental to building Debian packages.
- Android source code includes a hard dependency on the Java Developer Kit (JDK) 5.0 Update 12 or greater.

Run the command to install it:

```
# sudo apt-get install git-core gnupg flex bison gperf libSDL-dev libesd0-dev  
libwxgtk2.6-dev  
# sudo apt-get install build-essential zip curl libncurses5-dev zlib1g-dev
```

5.1.2 Install JDK 5.0

add this follow items in to Synaptic package source. (`sudo vi /etc/apt/sources.list`)

```
deb http://archive.ubuntu.com/ubuntu/ jaunty multiverse  
deb http://archive.ubuntu.com/ubuntu/ jaunty-updates multiverse
```

Update Synaptic package source

```
# sudo apt-get update
```

Install it

```
# sudo apt-get install sun-java5-jdk
```

5.2 Android compile

5.2.1 compile Android

copy the file `Android-2.1-Real6410-r1.tar.bz2` From CD to ubuntu system in PC.

Android-2.1-Real6410-r1.tar.bz2 position: **CD:\Android\code**

```
# mkdir Android
# cd Android
# cp -a /media/cdrom/android/code/Android-2.1-Real6410-r1.tar.bz2 .
# tar xvf Android-2.1-Real6410-r1.tar.bz2
# cd Android-2.1-Real6410-r1
```

Then we can get the `Android-2.1-Real6410-r1` folders in Android folder, run the command to compile it:

```
# sudo ./build_real6410
```

If there is no error, `android.tar.gz` should have been generated. you can get it in `Android-2.1-Real6410-r1/rootfs` folder.

5.2.2 Android module support

We provide the WIFI 、 Camera 、 JPEG 、 GPS support lib for android 2.1 with *.so file. Only the GPS module need configure when use this module.

In **Android-2.1-Real6410-r1/vendor/realarm/real6410** folder, it have the **libgps.so** file, it is the GPS file, and it have other file as that:

```
libgps-4800-debug.so
----GPS module for 4800 baudrate, it will print debug message when use it.
libgps-4800-release.so
----GPS module for 4800 baudrate, it dont't print debug message when use it.
libgps-9600-debug.so
----GPS module for 9600 baudrate, it will print debug message when use it.
libgps-9600-release.so
----GPS module for 9600 baudrate, it don't print debug message when use it.
```

the default **libgps.so** was came from the `libgps-9600-release.so`, so if you can run the GPS module in android, please replace the `libgps.so` file with the right file for yor GPS module.

Then rebuild the android again.

Chapter VI Android function Use

6.1 Ethernet use

The Android can link the web from the DM900A, we should set the net param, such as dns, ip, gateway.

Uncompress the Android system([android.tar.gz](#)), and edit it as follow:

A) set the dns

edit the `/init.rc` in Android file system folder, add the flow red word to this file

```
# Define TCP buffer sizes for various networks
# ReadMin, ReadInitial, ReadMax, WriteMin, WriteInitial, WriteMax,
setprop net.tcp.bufferize.default 4096,87380,110208,4096,16384,110208
setprop net.tcp.bufferize.wifi 4095,87380,110208,4096,16384,110208
setprop net.tcp.bufferize.umts 4094,87380,110208,4096,16384,110208
setprop net.tcp.bufferize.edge 4093,26280,35040,4096,16384,35040
setprop net.tcp.bufferize.gprs 4092,8760,11680,4096,8760,11680
setprop net.dns1 10.10.0.21
```

notice: the dns was set the address can be used in your net.

B) set ip and gateway.

edit the `/system/etc/init.real6410.sh` in Android file system folder as follow:

```
#!/system/bin/sh
ifconfig eth0 192.168.1.20 netmask 255.255.255.0 up
route add default gw 192.168.1.1 dev eth0
```

Notice: set the right ip and gw that can link to the web.

C) compress the folder again, and download it to the board.

6.2 USB adb in Android 2.1

A) When the board run and enter the Android2.1 system in LCD, click that button in LCD:

DevTools->Development Setting->Wait for debuuger.

B) Link the usb device to the PC by usb cable.

- If the PC was windows system, install the USB driver.

USB driver position: **CD:\tools\usb driver\USB android driver**

- If the PC was ubuntu system

Use the `lsusb` command first to check the connection between the PC and board.

```
# lsusb
Bus 001 Device 007: ID 18d1:0001
```

This is the android device, Pay attention to the first ID hex code part "18d1"

That id is used in the following steps to identify this device.

Create a new file (as root)

```
# sudo gedit /etc/udev/rules.d/50-android.rules
```

with the following content

```
SUBSYSTEM=="usb", SYSFS{idVendor}=="18d1", MODE="0666"
```

Make sure that the text after {idVendor} is the same as the device id found when using the lsusb command. Save and exit the editor.

Let the udev find the device:

```
# sudo service udev restart
```

kill the last server

```
# sudo /home/figo/ARM11/android/Android-2.1/out/host/linux-x86/sdk/
androidsdk_eng.figo_linux-x86/tools/adb kill-server
```

Notice the position is different with you PC, please use the right position.

Find the device

```
# sudo /home/figo/ARM11/android/Android-2.1/out/host/linux-x86/sdk/
androidsdk_eng.figo_linux-x86/tools/adb devices
```

If it is ok, it will print that:

```
# sudo /home/figo/ARM11/android/Android-2.1/out/host/linux-x86/sdk/
androidsdk_eng.figo_linux-x86/tools/adb devices
* daemon not running. starting it now *
* daemon started successfully *
List of devices attached
0123456789ABCDEF device
```

6.3 Camera use in android 2.1

When the system booting, if it print the follow message, it means the WIFI can be found in android.

```
[OV965X]ov965x_attach_adapter.
```

parent clock for camera: 266.666 MHz, divisor: 11
OV965X attached successfully

Then click "camera" in LCD, and run the command in serial

```
# logcat
```

It will print the message

```
.....  
D/CameraHardwareRealARM( 1920): *****  
D/CameraHardwareRealARM( 1920): *****RealARM*****  
D/CameraHardwareRealARM( 1920): *****Camera module for Real6410 *****  
D/CameraHardwareRealARM( 1920): *****Figo,sagres_2004@163.com *****
```

Then you can click the button in LCD to capture the pic, and the picture was stored in SD card, you should insert the SD card before you test the camera,

Not use the video for camera, at present it don't support.



6.4 WiFi use in android 2.1

When the system booting, if it print the follow message, it means the WIFI can be found in android.

```
mmc0: new SDIO card at address 0001  
libertas_sdio mmc0:0001:1: firmware: using built-in firmware sd8686_helper.bin  
libertas_sdio mmc0:0001:1: firmware: using built-in firmware sd8686.bin  
libertas: 00:22:43:73:26:bf, fw 9.70.3p24, cap 0x000003a3  
libertas: PREP_CMD: command 0x00a3 failed: 2  
libertas: PREP_CMD: command 0x00a3 failed: 2  
libertas: eth1: Marvell WLAN 802.11 adapter
```

Click "Settings->wireless->Airplane Mode" to remove the Airplane mode.

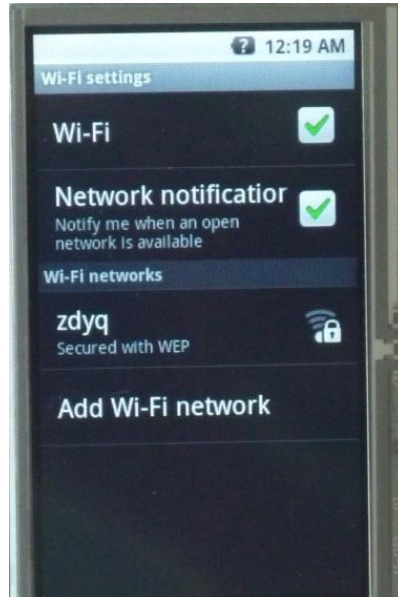
Then click "Settings->Wireless & network->WIFI" in LCD, and run the command in serial

```
# logcat
```

It will print the message

```
I/WifiHW ( 1940): *****  
I/WifiHW ( 1940): *****RealARM*****  
I/WifiHW ( 1940): *****WIFI module for Real6410 *****  
I/WifiHW ( 1940): *****Figo,sagres_2004@163.com *****  
I/WifiHW ( 1940): *****
```

Then the WiFi will search the AP, and you can surf the net by WiFi.



6.5 Switch horizontal and vertical screen

Press "menu" button until the screen Switch success.

6.6 Dynamic Wallpapers

click the touchscreen until it show the dialog , then click "**Wallpapers->Live Wallpapers**", then change it.

6.7 Install APK file in android 2.1

1) copy the apk file to SD card, and insert the sd card to the board, then run the program: eoeAppInstaller, it will find the apk file, then click to install it.

2) use adb

run the command in PC to install your apk

```
sudo /home/figo/ARM11/android/Android-2.1/out/host/linux-x86/sdk/androidsdk_  
eng.figo_linux-x86/tools/adb -d install your-apk
```

6.8 GPS module use

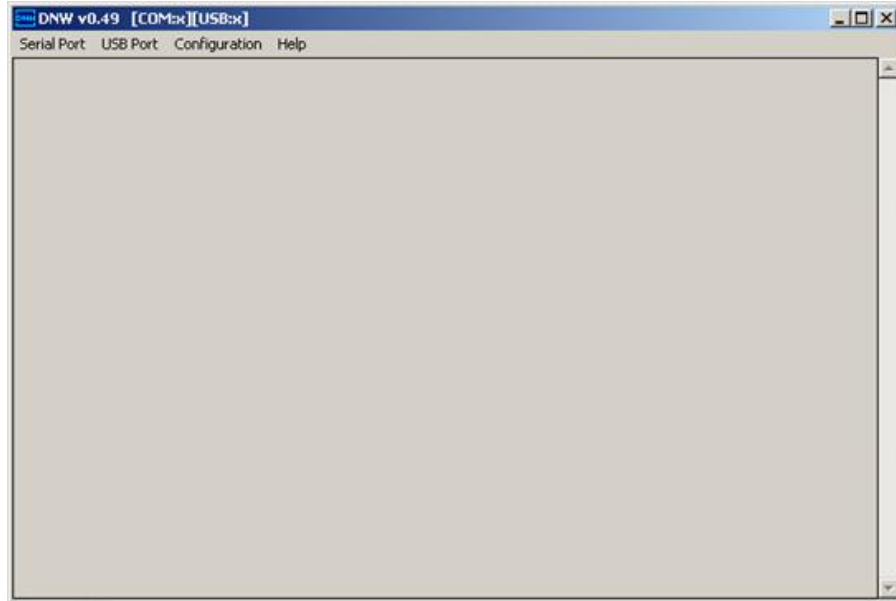
In LCD, Run the program: **GPS Status**

Then it will show:



Appendix A DNW software configuration

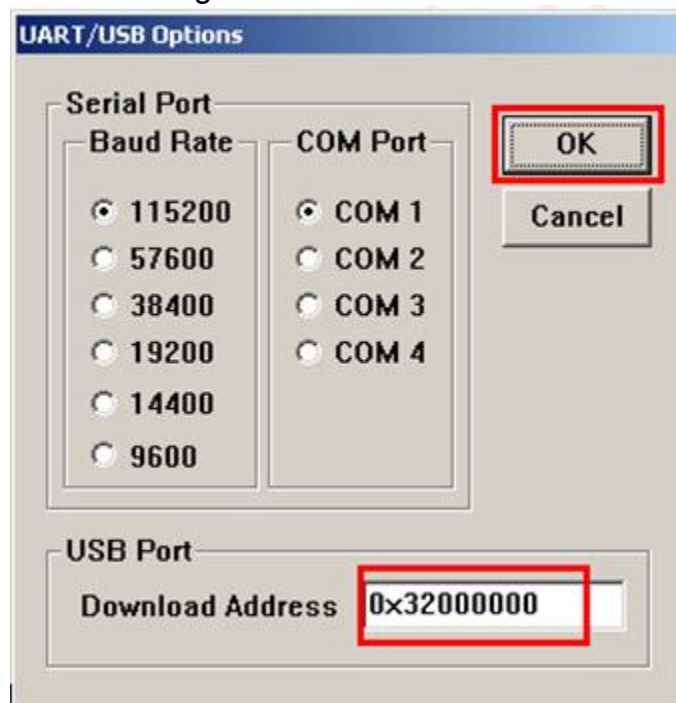
1. Find DNW software under directory **CD:\Tools\DNW.exe**. Double-click to open it:



2. Click "**Configuration -> Options**", it will open the "UART/USB Options" dialog.

- choose '**115200**' in '**Baud Rate**'
- choose '**COM1**' in '**COM Port**' (the COM1 means the serial number in PC)

click '**OK**' to finish the DNW configuration:



3. Then Click 'Serial Port->connect' to enable the DNW serial link.

Appendix B USB driver install

The following steps introduce how to install USB download-driver. The driver is located under the directory "**CD:/tools/usb driver/usb DNW driver/**":

1. Windows XP can recognize the new device automatically, it will show the follow message, choose "**No, not this time**", then click '**Next**'. (**figure B.1**)



2.

figure B.1

3. Then choose "**Install from a list or specific location[Advanced]**", then click "**Next**". (**figure B.2**)

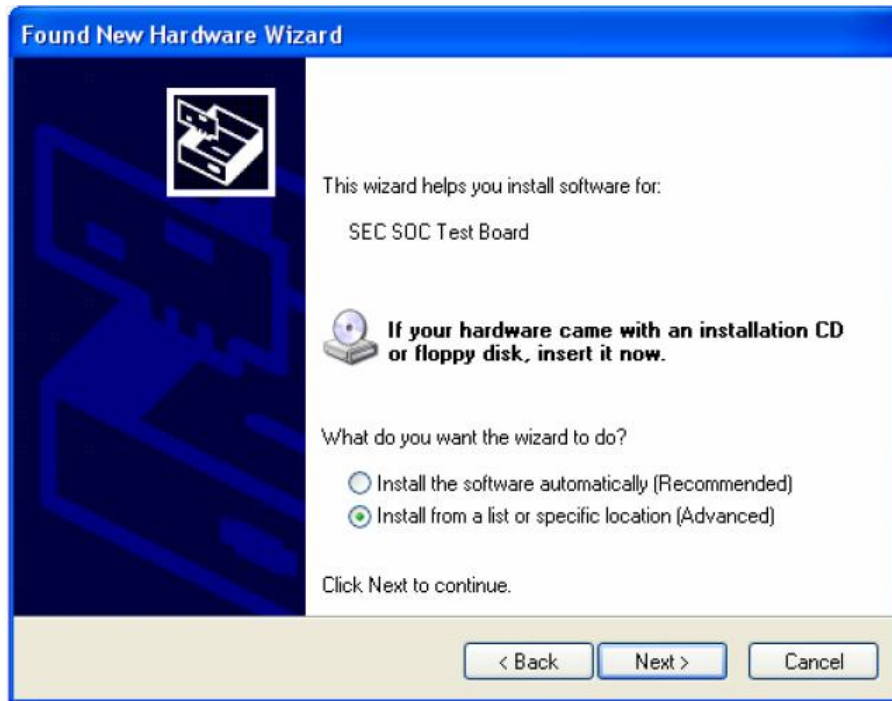


figure B.2

- Then in the option “**Search for the best driver in these locations**”, input the usb driver path “**CD:/tools/usb driver/usb DNW driver/**”, then click “**Next**”. (figure B.3)

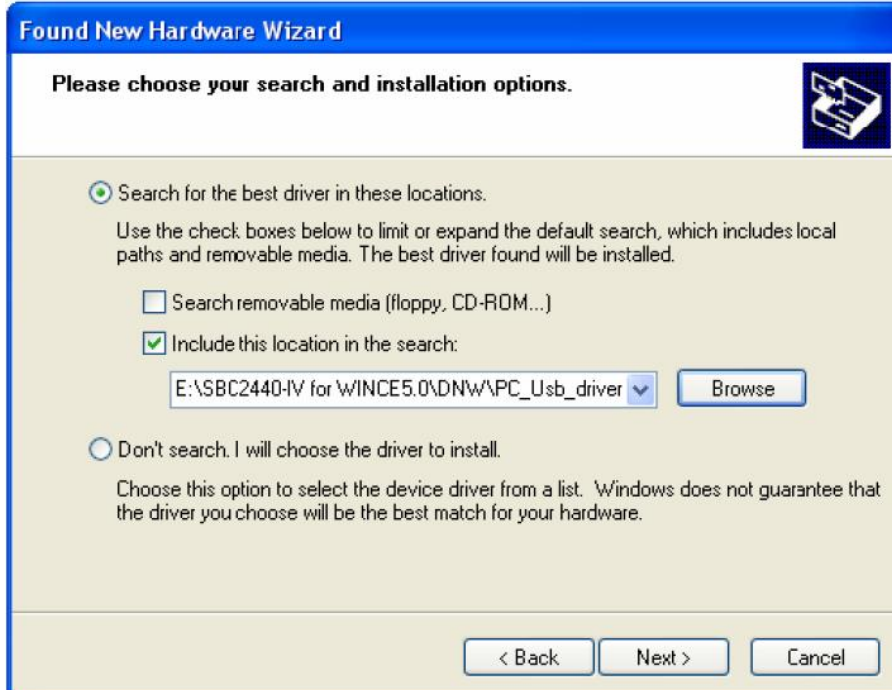
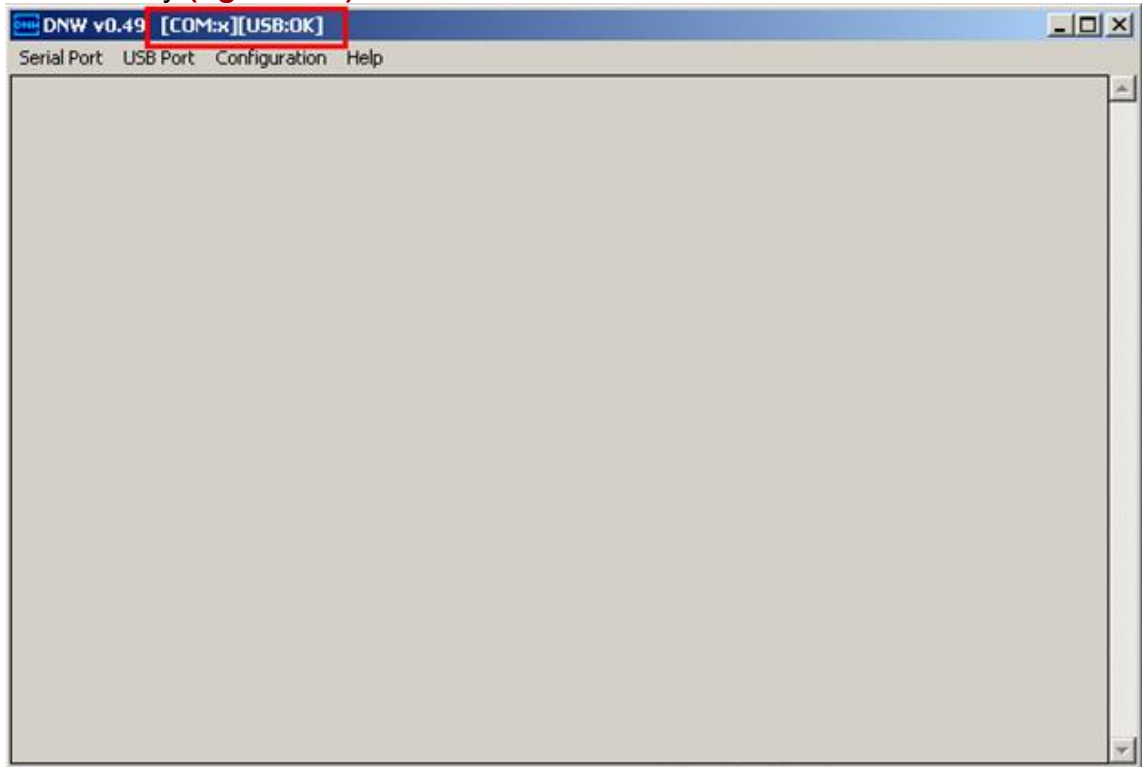


figure B.3

5. Then it will install the usb driver, maybe it have the different message in your PC, but you should make sure that the usb driver is this path: **“CD:/tools/usb driver/usb download driver/”**
6. If the DNW status show the USB:OK, it means that the driver will be installed successfully. **(figure B.4)**



Appendix C Cross compile tools install

C-1 Uncompress the tools

Copy the **arm-2008q3.tar.bz2** to /root/ folder
arm-2008q3.tar.bz2 position: CD:/linux/code

```
# cd /root
# sudo cp /media/cdrom/linux/code/arm-2008q3.tar.bz2 .
# sudo tar xvf arm-2008q3.tar.bz2
```

Then we can get the folder :**arm-2008q3** in the /root/

C-2 Add Path in your environment file

Modify your ~/.bashrc file to add a new path with editor (gedit or vi)

```
PATH=$PATH:/root/arm-2008q3/bin
```

To apply this change, login again or restart the .bashrc

```
# source .bashrc
```

C-3 Check the tool-chain path to see if it is set up correctly or not

```
~$ arm-none-linux-gnueabi-gcc -v
Using built-in specs.
Target: arm-none-linux-gnueabi
Configured with: /scratch/julian/lite-respin/linux/src/gcc-4.3/configure --build=i686-pc-linuxgnu
--host=i686-pc-linux-gnu --target=arm-none-linux-gnueabi --enable-threads --disablelibmudflap
--disable-libssp --disable-libstdcxx-pch --with-gnu-as --with-gnu-ld --enablelanguages=
c,c++ --enable-shared --enable-symvers=gnu --enable-__cxa_atexit --withpkgversion='
Sourcery G++ Lite 2008q3-72' --withbugurl=
https://support.codesourcery.com/GNUToolchain/ --disable-nls --prefix=/opt/codesourcery
--with-sysroot=/opt/codesourcery/arm-none-linux-gnueabi/libc --with-buildsysroot=/
scratch/julian/lite-respin/linux/install/arm-none-linux-gnueabi/libc --withgmp=/
scratch/julian/lite-respin/linux/obj/host-libs-2008q3-72-arm-none-linux-gnueabi-i686-pcReal6410
linux-gnu/usr --with-mpfr=/scratch/julian/lite-respin/linux/obj/host-libs-2008q3-72-arm-nonelinux-
gnueabi-i686-pc-linux-gnu/usr --disable-libgomp --enable-poison-system-directories --withbuild-
time-tools=/scratch/julian/lite-respin/linux/install/arm-none-linux-gnueabi/bin --with-buildtime-
tools=/scratch/julian/lite-respin/linux/install/arm-none-linux-gnueabi/bin
Thread model: posix
gcc version 4.3.2 (Sourcery G++ Lite 2008q3-72)
```

